SmishGuard: Leveraging Machine Learning and Natural Language Processing for Smishing Detection

Saleem Raja Abdul Samad[1], Pradeepa Ganesan[2], Justin Rajasekaran[3],
Madhubala Radhakrishnan[4], Hariraman Ammaippan[5], Vinodhini Ramamurthy[6]
College of Computing and Information Sciences, Information Technology Department,
University of Technology and Applied Sciences-Shinas, Sultanate of Oman[1,2,3.4,5]
Little Angel Institute, Karur, Tamil Nadu. India[6]

***Abstract*—SMS facilitates the transmission of concise text messages between mobile phone users, serving a range of functions in personal and business domains such as appointment confirmation, authentication, alerts, notifications, and banking updates. It plays a vital role in daily communication due to its accessibility, reliability, and compatibility. However, SMS unintentionally generates an environment where smishing can occur. This is because SMS is extensively available and reliable. Smishing attackers exploit this trust to trick victims into divulging sensitive information or performing malicious actions. Early detection saves users from being victimized. Researchers introduced different methods for accurately detecting smishing attacks. Machine Learning models, coupled with Language Processing methods, are promising approaches for combating the escalating menace of SMS phishing attacks by analyzing large datasets of SMS messages to differentiate between legitimate and fraudulent messages. This paper presents two methods (SmishGaurd) to detect smishing attacks that leverage machine learning models and language processing techniques. The results indicate that TF-IDF with the LDA method outperforms Weight Average Word2Vec in precision and F1-Score, and Random Forest and Extreme Gradient Boosting demonstrate higher accuracy.**

***Keywords*—Smishing; phishing; SMS; machine learning; natural language processing; TF-IDF**

I. INTRODUCTION

SMS (Short Message Service) is a pervasive mode of communication that spans geographic boundaries and device types in the digital world. It delivers information, warnings, and notifications quickly and reliably, making it essential for personal and professional interactions. However, this convenience raises the risk of smishing. Smishing, a malicious combination of "SMS" and "phishing," represents a grievous and pervasive threat to cybersecurity [1]. It is a particularly effective vector for cybercriminals due to the prevalence of mobile phones and the inherent trust in text messages, that causes serious harm to both individuals and organizations. Many victims are tricked into giving over personal information, clicking on harmful links, or unintentionally installing malware on their mobile devices, compromising their financial security, privacy, and digital identity [2]. The compromise of sensitive personal information can lead to identity theft, unauthorized account access, and additional intrusions. Smishing is especially dangerous because attackers are constantly changing their strategies, and with the low technical barrier to entry, attackers of all ability levels can participate, making this danger widespread. They use social engineering and psychological tricks to make text messages that look real and trustworthy. Even the most cautious people can be tricked by urgent messages posing as from banks, governments, or well-known businesses, forcing them to take steps that would benefit the attackers [3]. Additionally, when personal and financial information is stolen, it makes people more likely to be victims of hacking in the future, since attackers can use this information for damaging purposes.

In this dynamic environment, machine learning (ML) arises as a formidable ally in the defense against smishing attacks [4]. By their very nature, machine learning algorithms excel at pattern recognition, allowing for the detection of nuanced clues within text messages that reveal fraudulent intent. Most importantly, machine learning systems always learn from new data and adapt to the changing strategies used by smishing attackers. This adaptability is crucial because smishing attacks evolve to avoid detection and capitalize on emergent trends. Furthermore, machine learning scales well, enabling real-time analysis of enormous amounts of SMS messages across large mobile networks, enabling proactive detection and prevention. One of the main goals of machine learning in smishing detection is to reduce false positives, which prevents valid SMS messages from being inadvertently tagged as suspicious and protects the integrity of communication channels. Machine Learning with Natural Language Processing (NLP) methods enhance the detection of smishing attacks. ML-NLP models excel at identifying linguistic anomalies and patterns within SMS messages, distinguishing smishing-specific keywords, phrases, and grammatical inconsistencies [5].

Researchers introduced several smishing detection strategies using deep and machine learning techniques. Most methodologies use machine learning or deep learning models to extract features from textual content for classification. Generating features from textual data, such as counting the number of words or special characters in a document, provides structural insights and facilitates data representation. However, these features are not concerned with the context or semantics of the text but rather with its structural characteristics. Sometimes, the generated features may not optimize the potential of the dataset. Particularly in smishing detection, the context of the textual content is more critical than structural characteristics. By capturing the relationships

between words and contextual meaning, NLP methods for fully vectorizing text, such as TF-IDF (Term Frequency Inverse Document Frequency) or word embeddings, provide a richer grasp of the text's semantic content and make complicated analysis and machine learning tasks possible [6]. This paper introduces two novel methods (SmishGuard), which use TF-IDF with LDA (Latent Dirichlet Allocation) topic proportion score and Average Word2vec with TF-IDF score as weight for smishing detection and compares the performances. The results show that algorithms using random forest and extreme gradient boosting attain higher accuracy.

Contribution of the work:

- Two novel methods (SmishGuard) for smishing detection.
- The first method combines TF-IDF with LDA topic proportion scores to improve smishing detection by providing a comprehensive view of SMS messages that captures both term-level and underlying topics.
- The second method uses Average Word2Vec with TF-IDF scores as weights for smishing detection capitalizing on the strengths of both approaches, capturing semantic information and term importance.
- Proposed methods decrease false positives and enhance cybersecurity by enabling more precise and context-aware detection.
- Experimental evaluation and Performance Comparisons.

The research work's sections are arranged as follows. Section I delineates the problem's significance and outlines the proposed research works. The background is in Section II. In Section III, related works are highlighted. Section IV describes the proposed methodologies in detail. Experimental results are explained in Section V. The paper concludes with Section VI.

II. BACKGROUND

Smishing attacks, a form of phishing that utilizes SMS or text messaging, pose a serious cybersecurity risk. In these attacks, scammers typically send false communications to victims to coerce them into disclosing personal information, opening malicious links, or downloading dangerous files. Machine learning analyses content, sender information, and message context to detect smishing using natural language processing methods. Through training models on labeled datasets that include examples of both smishing and real messages, machine learning systems can identify patterns, language indications, and typical behavioral oddities. This proactive approach enhances the capability to automatically identify and flag suspicious messages, thereby protecting users from smishing frauds and boosting the security of mobile communication channels. This section describes the key technologies utilized in the proposed methods.

- TF-IDF Vectorizer: Using vectorizers, the unstructured text data are transformed into a numerical representation that machine learning algorithms can process. They convert text to numerical vectors. TF-IDF vectorizer creates a matrix where each document is a row, and each distinct word is a column from a set of text documents. A term's TF-IDF score is calculated by multiplying TF and IDF. TF counts how frequently a term appears in a document, while IDF measures a word's rarity across the corpus [7].
- Latent Dirichlet Allocation (LDA): LDA plays a key role in detecting smishing (SMS phishing) by revealing hidden themes and topics in SMS communications. It enables the identification of underlying linguistic patterns frequently observed in fraudulent messages, enabling the classification of incoming texts as either legitimate or potentially malevolent. By leveraging LDA, smishing detection systems acquire the ability to distinguish context and content nuances, making them more adept at recognizing attacker's deceptive techniques [8] [9].
- Average Wor2Vec Word Embedding: It expands on Word2Vec word embeddings by providing a method for representing entire sentences or documents as dense vectors. Average Word2Vec computes the vector representation of a sentence by calculating the average of the word vectors in it rather than considering each word separately. The approach involves converting each sentence word to its Word2Vec form and determining the mean of these vectors. Thus, the whole phrase is a short, fixed-length vector that captures its meaning. Average Word2Vec efficiently represents sentences while preserving semantic links and context from the Word2Vec model. It helps machine learning algorithms understand sentences and documents of different lengths by providing a constant vector dimension [10].
- Machine Learning Algorithms (ML): In detecting smishing messages, machine learning models are quite effective [11]. They are essential in recognizing and classifying smishing messages with dangerous or harmful information, which helps to improve cyber security and shield users from potential risks. Most of the research work utilizes different machine learning algorithms for smishing detection using SMS text, such as Logistic Regression (LogR), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB) Gaussian Naive Bayes (GNB), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), Ada Boosting (AB), Extra Tree (ET) and Extreme Gradient Boosting (XGB).

III. RELATED WORKS

Detecting and preventing smishing attacks is essential for safeguarding individuals and organizations from potential harm. Effective methods for detecting smishing attempts have been developed using machine learning and deep learning (DL) approaches. The most recent findings from research conducted in this area are presented in this section.

Boukari et al. [12] proposed a fraud detection system that utilizes the TF-IDF method to convert SMS text into a vectorized representation. A dataset of 5000 emails was used

for smishing. The importance of each word in the SMS is determined using the TF-IDF algorithm. The experiment's findings indicate a high level of accuracy, with the RF algorithm achieving an accuracy of 98.15% and the NB approach achieving an accuracy of 90.59%. Mishra et al. [13] developed an efficient smishing detection system using an artificial neural network. The dataset utilized in the experiment consisted of 5858 messages, with 538 classifieds as smishing messages and 5320 as valid messages. Seven distinctive features were extracted from the dataset. The neural network was implemented with seven features, including email, URL, phone number, etc. The performance results for NB, DT, and Neural Network were 96.29 %, 93.40 %, and 97.40 %, respectively. Ulfath et al. [14] employed the N-grams and TF-IDF techniques for feature extraction and the statistical feature selection method. The vocabulary only includes the top 10,000 terms across the SMS dataset. The dataset was obtained from UCI's machine learning library. Five machine learning algorithms, XGB, DT, RF, SVM, and AB were employed. The outcome reveals that the SVM classifier obtains a higher accuracy of 98.39%. Smishing messages from various Internet sources were collected by Mishra et al. [15]. There are 638 smishing, 489 spam, and 4844 ham messages in the 5971 samples. Finally, five features were extracted and used in the experiment. ML algorithms such as NB, RF, and DT were employed. The performance outcomes for NB were 94.06%, RF 94.64%, and DT 93.96%. Maqsood et al. [16] used both ML and DL classifiers to detect spam SMS. The dataset was obtained via Kaggle, (SMS Spam Collection Dataset). The process of extracting features was conducted via the Bag-of-Words and TF-IDF methodologies. The gathered features served as input for three distinct machine learning models and CNN in deep learning. The SVM demonstrated a high level of accuracy, outperforming the other models with a performance of 99.6% for spam SMS. Ramanujam et al. [17] compiled a multiple languages SMS dataset, which encompasses both spam and non-spam messages in English and four distinct Indian languages. The SMS data contains 2,757 ham and 525 spam messages. A deep learning hybrid model (CNN-LSTM) has been proposed without feature engineering. The model exhibits a 97.7% accuracy rate. Jain et al. [18] presented a method to classify messages as smishing messages. The model was trained using multiple machine learning techniques with the Almeida spam data collection of 5169 messages, 362 of which are smishing and 4817 non-smishing. In addition, the model's accuracy has been determined for across datasets. The voting classifier that integrates KNN, RF, and ET Classifier (ETC) achieves an accuracy of 99.03% and a precision of 98.94%. Using machine learning, Mishra et al. [19] extracted the five most effective text message features for smishing detection. The dataset used for smishing detection contains 5858 text messages, 538 of which are smishing and 5320 valid. For the experimental results, RF, NB, DT, and Backpropagation Algorithm were used. The Backpropagation Algorithm outperformed the competition with a 97.93% accuracy rate. Sjarif et al. [20] used an algorithm incorporating TF-IDF and machine learning algorithms to identify SMS spam messages. The UCI Repository provided the dataset. The collection includes 5,574 English raw text messages categorized as ham or spam. Of these messages, 4,827 are classified as ham and the remaining 747 as spam. Five ML algorithms were used combined with the TF-IDF method for experimental purposes. The TF-IDF and RF combination produced an impressive 97.50% accuracy rate.

In existing research, feature extraction and vectorization from the SMS dataset are prioritized. Understanding the content of the message is essential for improved classification. The existing related works summary is shown in Table I.

TABLE I. RELATED WORKS SUMMARY

Author	Dataset	Feature generation method	Accuracy	Issues
Boukari et al. [12]	Smishing dataset 5000 messages	TF-IDF Vectorizer	Naïve Bayes: 90.59% Random Forest: 98.15%	Contextualization of text is not included.
Mishra et al. [13]	Smishing dataset Smishing messages=538 Legitimate messages=5320	Feature Extraction	Naïve Bayes: 96.29 Decision Tree: 93.40 Neural Network:97.40	A limited number of features. The system does not fully utilize the dataset.
Ulfath et al. [14]	UCI Dataset (SMS-Spam Collection) Spam messages=1197 Legitimate messages=4377	TF-IDF Vectorizer	Support vector machine: 98.39	Contextualization of text is not included.
Mishra et al. [15]	Smishing dataset Smishing messages=638, Spam messages=489 Legitimate messages=4844	Feature Extraction	Nave Bayes: 94.06, Random Forest: 94.64, Decision Tree: 93.96	The system is based on 5 features. The system does not fully utilize the dataset.
Maqsood et al. [16]	Kaggle- SMS Spam Collection Dataset. Spam messages=750, Legitimate messages=4250	Count (BoW) and TF-IDF Vectorizer	Support vector machine: 99.6	Contextualization of text is not included.
Ramanujam et al. [17]	Multilingual SMS Dataset. Legitimate messages=2,757, Spam messages=525	--	CNN-LSTM model: 97.7	Ne specific feature engineering was adopted.
Jain et al. [18]	Almeida spam data set. Spam Messages=362, Legitimate Messages=4817	TF-IDF Vectorizer	Voting classifier with KNN, RF, and ET Classifier (ETC) =99.03	Time-consuming method
Mishra et al. [19]	Smishing dataset Smishing messages=538 Legitimate messages=5320	Feature Extraction	Backpropagation Algorithm: 97.93	A limited number of features. The system does not fully utilize the dataset.
Sjarif et al. [20]	UCI Dataset Legitimate message=4,827 Spam messages= 747	TF-IDF Vectorizer	RF:97.50	Contextualization of text is not included.

IV. PROPOSED METHODS

Our proposed methods (SmishGuard) include two different systems for smishing (SMS phishing) detection that employ ML and NLP methods to improve mobile communications security. Most of the existing methodologies extract features from textual content for classification using ML or DL models. Generating features from textual data, such as counting the number of words or special characters in a document, provides structural insights and facilitates the representation of data. However, these features are not concerned with the context or semantics of the text, but rather with its structural characteristics. Sometimes, the generated features may not optimize the potential of the dataset. The primary goal of the proposed system is to maximize the utilization of the dataset through the implementation of several natural language processing methods, including TF-IDF with LDA and average Word2Vec with TF-IDF. NLP methods fully vectorizing text, such as TF-IDF or word embeddings, provide a richer grasp of the text's semantic content and make complicated analysis and machine learning tasks feasible. The outcomes of both methods exhibit a higher level of performance in comparison to the currently available methods. The process flow of the proposed system is shown visually in Fig. 1.

A. Dataset

For experiments, two distinct data sources are combined. Table II provides information about the data sources. Combining data from multiple sources necessitates the elimination of duplicate entries from the dataset. After removing duplicates, the final dataset used for our experiment is presented in Table III.

TABLE II. DATA SOURCES

Data Set	Features
SMS Phishing Dataset [21]	Smishing =1127 Legitimate messages=4844.
SMS Spam Collection [22]	Legitimate Messages=4825 Spam messages=747

TABLE III. COMBINED DATASET

Data Set	Features
SMS_PHISH	Smishing Messages =1627 Legitimate messages=5634

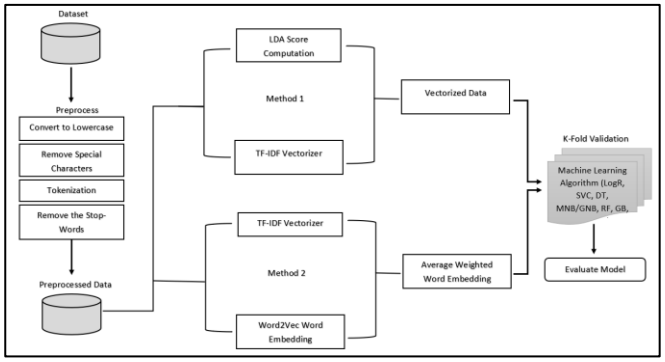


Fig. 1. Process flow of the proposed system.

B. Preprocessing

Text preprocessing is a crucial initial stage for Natural Language Processing and Machine Learning tasks that involve textual data. Effective text preprocessing is essential for maintaining data integrity and eliminating irrelevant information, enabling NLP and ML models to concentrate on extracting valuable insights and discerning patterns from textual data [23] [24]. Common procedures include lowercasing, tokenization, stop word elimination, and special character handling, etc. Algorithm 1 describes the preprocessing procedures.

Algorithm 1: Processing

1. Input: Raw Text
2. Output: Preprocessed Text
3. Import required libraries (nltk for natural language processing and re for regular expressions).
4. Load the raw text data.
5. Convert text to lowercase for consistency.
6. Remove any special characters, punctuation, and numerical values using regular expressions.
7. Break the text down into individual words or tokens.
8. Eliminate stop words as they do not contribute much to the meaning.
9. Join the tokens back into a preprocessed text string.

C. Method 1: TF-IDF Vectorization with LDA Topic Proportion Score (TF-IDF-LDA)

Combining TF-IDF vectorization and LDA text proportion scores is an effective approach. Through TF-IDF, text messages are transformed into numerical feature vectors that highlight the significance of phrases in each message, allowing machine learning algorithms to identify patterns suggestive of attempted smishing. Along with LDA topic proportion scores (The number of topics is set to 2 for our experiment) improve the identification accuracy of fraudulent messages. These scores provide a numerical representation of the subjects present in each message by applying LDA to SMS content. Smishing attempts often use certain language patterns or themes. LDA can reveal hidden motifs in smishing efforts, helping models distinguish between authentic and suspect communications. High proportions of topics relating to fraud, urgency, or deceptive content can serve as strong indicators of smishing, enabling the development of robust detection systems that protect users from phishing threats concealed within text messages.

Term Frequency (TF) for a word in a document [25]

$$\text{TF}(w, d) = \frac{n_{w,d}}{N_d}$$

$n_{w,d}$ = The frequency of occurrences of the term w in document d.

N_d = Total word count in document d.

Inverse Document Frequency (IDF) for a word in the corpus:

$$\text{IDF}(w) = log\left(\frac{\text{N}}{n_w}\right)$$

N= Number of documents contained in the corpus

n_w = Count of documents that contain the word w

TF-IDF Score for a word in the document is obtained by multiplying TF and IDF.

Topic Proportion Calculation for a Document [26]

For a given document D and topic T:

$$P(T|D) = \frac{P(T).P(D|T)}{P(D)}$$

where:

P(T|D) is the probability (proportion) of topic T in document D.

P(T) is the topic's prior probability over the entire corpus. It represents the proportion of documents within the entire dataset that are assigned to topic T.

P(D|T) is the probability of document D being generated from topic T. It determines the probability that the words in document D are generated by topic T.

P(D) is the probability of observing document D in the corpus.

D. Method 2: Weighted Average Word Embedding Generation

After preparing the raw textual contents, preprocessed data is prepared for the experiment. To determine the weight of words within a corpus through the utilization of TF-IDF vectorization, the initial step involves the computation of TF-IDF scores for every individual word encompassed within the entirety of the corpus. The TF-IDF metric quantifies the significance of a word within a particular document by considering its frequency in that document and inversely comparing it to its frequency across all documents. After obtaining these TF-IDF scores, the next step is to compute the weighted average Word2Vec embedding for machine learning using these scores as weights. The Word2Vec model is utilized to represent words as continuous vectors that capture semantic relationships. To calculate the weighted average Word2Vec embedding for a document, the vector of Word2Vec for each word is multiplied by its corresponding TF-IDF score. The result is a weighted vector. Then the sum of all weighted vectors of the document is divided by the sum of all weights of a document. Every document in the corpus finally gets a single weighted average word vector. This weighted average Word2Vec representation includes word importance and semantic context for downstream machine-learning tasks. The entire process is presented in Algorithm 2.

Weighted Average Word2Vec Computation for a document 'd' [27]

Multiply each word's Word2Vec vector by its corresponding TF-IDF score. The weighted average word embedding is calculated using the equation below:

Weighted Average Word2Vec (d) =

$$\frac{\sum_{w \in d} \left(TF - IDF_{(w,d)} \times Word2Vec\,(w) \right)}{\sum_{w \in d} \left(TF - IDF_{(w,d)} \right)}$$

where, w represents the word and d represents the document.

Where n is the number of words in a document.

Algorithm 2: Weighted Average Word2Vec

Input: TF-IDF matrix, Word2Vec embeddings

Output: Weighted Average Word2Vec representation for a document.

1. Multiply each word's Word2Vec vector by its TF-IDF score in the matrix.
2. Calculate the weighted average word embeddings for each of the document's words.

E. Training and Testing

Finally, K-fold validation trains and tests machine learning models using vectorized data/ word embedding. K-fold cross-validation is a widely employed technique utilized for the assessment of a model's performance and the mitigation of overfitting [28]. In this technique, the dataset is partitioned into K subsets, sometimes referred to as "folds," which are approximately of similar size. K-fold cross-validation trains and evaluates the model K times. For each iteration, one-fold is marked as the validation set, and K-1 folds are used for training. The performance metric is determined by calculating the average of K evaluation results.

V. Experiments and Results

The experiments employ Jupyter Notebook and Python's sklearn package on Windows 10. Accuracy, precision, recall, and the f1-score are performance parameters that are considered. Table IV presents the experimental result of the TF-IDF Vectorization with the LDA Topic Proportion Score (TF-IDF-LDA) method. Fig. 2 shows the ROC AUC curve.

The outcome demonstrates that the methods used in Random Forest and Extreme Gradient Boosting attains accuracy levels of 98.42% and 98.47%, respectively. Additionally, they obtained respective F1-Scores of 96.52% and 96.30%.

Table V presents the parameter details of the Word2Vec model. Table VI presents the experimental result of the weighted average word2vec (WAW2Vec) method. Fig. 3 shows the ROC AUC curve for the Random Forest and Extreme Gradient Boosting Algorithm.

TABLE IV. Performance of TF-IDF-LDA Method

ML Algorithm	Accuracy %	Precision %	Recall %	F1-Score %
LogR	96.97	95.26	91.03	93.07
SVC	98.33	98.33	94.16	96.17
MNB	97.52	98.58	90.22	94.19
DT	97.42	94.60	93.85	94.20
RF	**98.42**	**99.53**	**93.36**	**96.30**
GB	97.08	95.85	90.90	93.27
AB	97.22	94.97	92.50	93.70
XGB	**98.47**	**97.91**	**95.20**	**96.52**

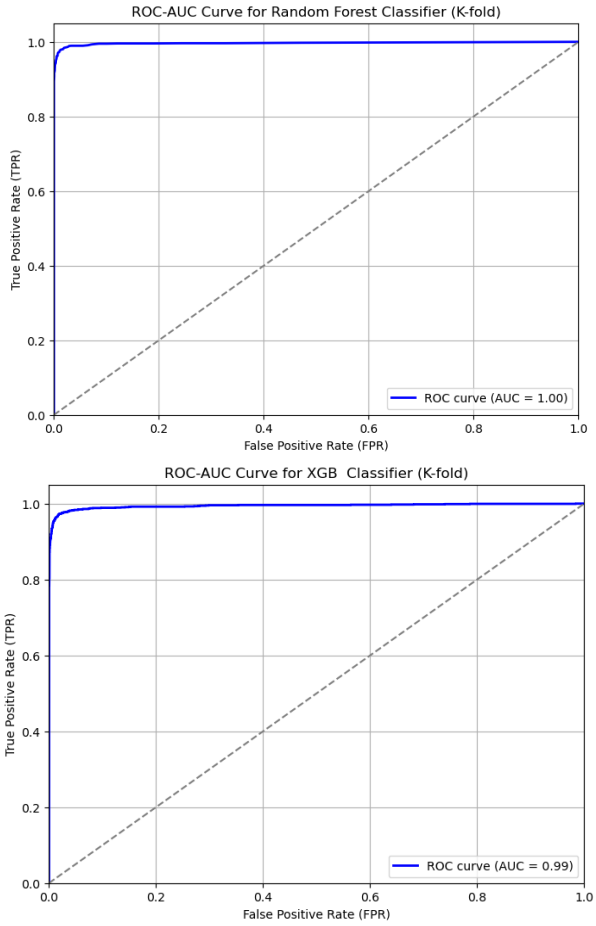


Fig. 2. ROC-AUC curve for TF-IDF-LDA method.

TABLE V. PARAMETER OF WORD2VEC MODEL

Parameter	Value
Vector Size	100
Window	5
Min_count	1
Sg	0

TABLE VI. PERFORMANCE OF WAW2VEC METHOD

ML Algorithm	Accuracy	Precision	Recall	F1-Score
LogR	86.08	85.35	90.07	87.34
SVC	91.23	91.23	92.55	91.78
GNB	82.83	87.58	79.18	82.53
DT	94.13	85.61	88.62	87.05
RF	**96.09**	**92.50**	**89.79**	**91.07**
GB	94.33	88.67	85.61	87.09
AB	93.04	84.94	83.83	84.34
XGB	**96.21**	**92.23**	**90.71**	**91.42**

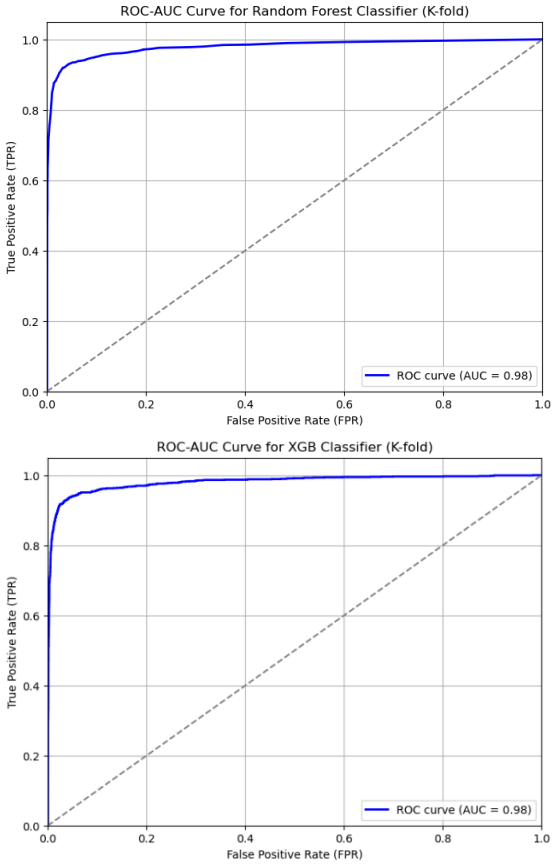
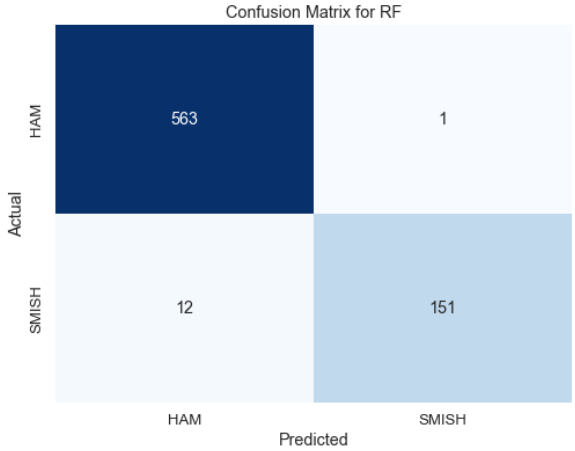


Fig. 3. ROC-AUC curve for WAW2Vec method.

The results indicate that Random Forest and Extreme Gradient Boosting obtained an accuracy of 96.09% and 96.22%, respectively. They also obtained respective F1-Scores of 91.07% and 91.42%.

Confusion matrix is crucial to classification model evaluation as it provides extensive insights into model performance. It divides the predictions and actual outcomes of the model into four categories: true positives, true negatives, false positives, and false negatives. Fig. 4 and Fig. 5 illustrate the performance of the proposed model by displaying fewer false positives and false negatives.



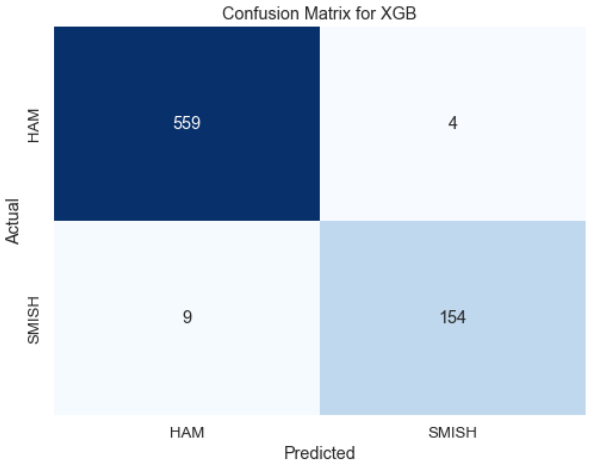


Fig. 4. Confusion matrix for TF-IDF-LDA method.

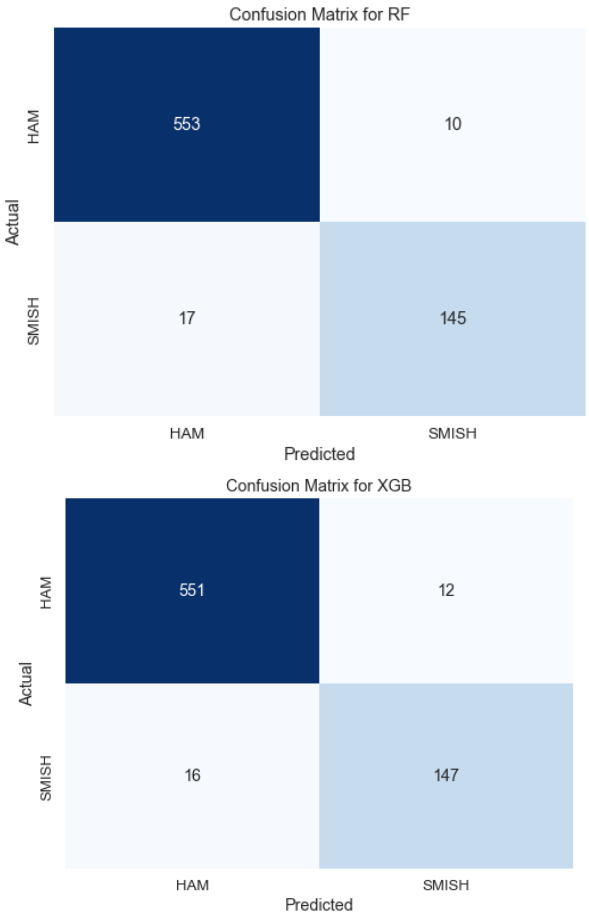


Fig. 5. Confusion matrix for WAW2Vec method.

The results demonstrate that the TF-IDF with LDA approach (Method-1) exhibits superior accuracy and F1-Score compared to the Weight Average Word2Vec method (Method-2). However, both methods produce better outcomes than existing methods as shown in Fig. 6.

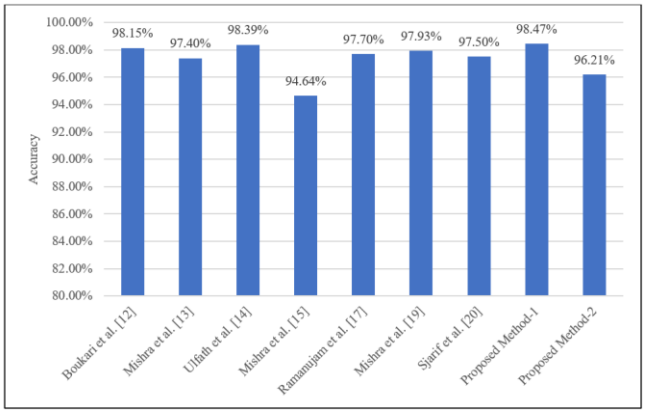


Fig. 6. Performance comparison (accuracy).

VI. Conclusion

Cybercriminals are increasingly using smishing, also known as SMS phishing, as a sneaky way to trick and deceive people and businesses. The expanding usage of mobile devices and text messaging for communication and transactions has increased the risk of smishing attacks. Attackers typically imitate reputable organizations to trick victims into disclosing critical information or committing crimes. So, smishing detection systems are vital for maintaining the security of vital infrastructure, protecting private and financial information, and maintaining public confidence in digital communication. Using ML and NLP methods for Smishing Detection is a promising way to counter the growing menace of SMS phishing attempts. This approach improves smishing detection by analyzing message content and contextual information.

This paper presents two methods based on ML algorithm and NLP methods for smishing detection. The results indicate that the TF-IDF with LDA approach (Method-1) achieves greater precision and F1-Score than the Weight Average Word2Vec technique (Method-2). Nevertheless, both methodologies yield outcomes that exhibit enhancements compared to current approaches. This empirical evidence is crucial to cybersecurity research since it refines methods and guides future study. Proposed method enhances the overall security of digital communication and transactions by contributing to the development of effective tools to counter evolving cyber threats.

Despite the potential of the proposed smishing detection system to mitigate the exponential growth of the smishing threat, the current system's capabilities are restricted to smishing messages in the English language. The task of managing diverse languages is progressively becoming more difficult.

References

[1] A.Kanaoka and T.Isohara, “Beyond Mobile Devices: A Cross-Device Solution for Smishing Detection and Prevention”, USENIX Symposium on Usable Privacy and Security, pp. 6–8, 2023.

[2] Smishing: https://dgc.org/en/smishing/ . (Last access: 20/9/2023).

[3] Malware: https://www.malwarebytes.com/what-is-smishing. (Last access: 20/9/2023).

[4] A.Mahmood and S.Hameed, "Review of Smishing Detection Via Machine Learning," Iraqi Journal of Science, 64(8), pp. 4244–4259, 2023. https://doi.org/10.24996/ijs.2023.64.8.42.

[5] A.Alhogail and A.Alsabih, "Applying machine learning and natural language processing to detect phishing email," Computers and Security, vol.1,10, 2021. https://doi.org/10.1016/j.cose.2021.102414.

[6] B.Sharma and P.Singh, "An improved anti-phishing model utilizing TF-IDF and AdaBoost," Concurrency Computation Practice and Experience. 34(26):e7287, 2022. https://doi.org/10.1002/cpe.7287.

[7] Mukesh, "TF-IDF Vectorizer scikit-learn," https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a. (Last access: 20/9/2023).

[8] E.Gualberto, R.Sousa, T.Vieira, J.Costa and C.Duque, "From Feature Engineering and Topics Models to Enhanced Prediction Rates in Phishing Detection," in IEEE Access, vol. 8, pp. 76368-76385, 2020. doi: 10.1109/ACCESS.2020.2989126.

[9] S.Lee, S.Kim, S.Lee, J.Choi, H.Yoon, D.Lee and J.Lee, "LARGen: Automatic Signature Generation for Malwares Using Latent Dirichlet Allocation," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 5, pp. 771-783, 2018. doi: 10.1109/TDSC.2016.2609907.

[10] Word embedding: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/. (Last access: 20/9/2023).

[11] J.Nabi, "Machine Learning Fundamentals," 2018 https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916. (Last access: 20/9/2023).

[12] B.Boukari, A.Ravi and M.Msahli, "Machine Learning detection for SMiShing frauds," IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), 2021.

[13] S.Mishra and D.Soni, "Implementation of 'Smishing Detector: An Efficient Model for Smishing Detection Using Neural Network," SN Computer Science, Springer. 2022. https://doi.org/10.1007/s42979-022-01078-0.

[14] R.Ulfath, I.Sarker, M.Chowdhury and M.Hammoudeh, "Detecting Smishing Attacks Using Feature Extraction and Classification Techniques," Proceedings of the International Conference on Big Data,IoT, and Machine Learning, Lecture Notes on Data Engineering and Communications Technologies, 95, 2022. https://doi.org/10.1007/978-981-16-6636-0_51

[15] S.Mishra and D.Soni, "SMS Phishing Dataset for Machine Learning and Pattern Recognition," Proceedings of 14th International Conference on Soft Computing and Pattern Recognition, Lecture Notes in Networks and Systems, 648, pp. 597–604, 2023. https://doi.org/10.1007/978-3-031-27524-1_57

[16] U.Maqsood, S.Rehman, T.Ali, K.Mahmood , T.Alsaedi and M.Kundi, "An Intelligent Framework Based on Deep Learning for SMS and e-mail Spam Detection," Applied Computational Intelligence and Soft Computing, 2023. https://doi.org/10.1155/2023/6648970.

[17] E.Ramanujam, K.Shankar and A.Sharma, "Multi-lingual Spam SMS detection using a hybrid deep learning technique," IEEE Silchar Subsection Conference (SILCON), pp. 1-6, 2022. doi: https://doi.org/10.1109/SILCON55242.2022.10028936

[18] A.K.Jain , B.B Gupta, K. Kaur, P.Bhutani, A.Almomani and W.Alhalabi, "A content and URL analysis-based efficient approach to detect smishing SMS in intelligent systems," International Journal of Intelligent Systems. 2022. https://doi.org/10.1002/int.23035

[19] S.Mishra and D.Soni, "DSmishSMS-A System to Detect Smishing SMS," Neural Computing and Applications. Springer. 35, pp. 4975–4992, 2023. https://doi.org/10.1007/s00521-021-06305-y.

[20] N.Sjarif, N.Azmi, S.Chuprat, H.Sarkan, Y.Yahya and S.Sam, "SMS Spam Message Detection using Term Frequency-Inverse Document Frequency and Random Forest Algorithm," Procedia Computer Science, vol. 161, pp. 509-515, 2019. https://doi.org/10.1016/j.procs.2019.11.150.

[21] S.Mishra and D.Soni, https://data.mendeley.com/datasets/f45bkkt8pr/1. (Last access: 20/9/2023).

[22] UCI Dataset: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset. (Last access: 20/9/2023).

[23] Y.Kerner, D.Miller and Y.Yigal, " The influence of preprocessing on text classification using a bag-of-words representation," PLoS One. 1;15(5):e0232525, 2020. doi: 10.1371/journal.pone.0232525.

[24] A.Saleem, B.Sundarvadivazhagan, R.Vijayarangan and S.Veeramani, "Malicious Webpage Classification Based on Web Content Features using Machine Learning and Deep Learning, International Conference on Green Energy," Computing and Sustainable Technology (GECOST), Miri Sarawak, Malaysia, pp. 314-319, 2022. doi: 10.1109/GECOST55694.2022.10010386. [Included in IEEE Xplore]

[25] S.W.Kim and J.M.Gil, "Research paper classification systems based on TF-IDF and LDA schemes," Human-centric Computing and Information Sciences, vol. 9, no. 1, pp 1–21, 2019. https://doi.org/10.1186/s13673-019-0192-7

[26] J.Gan and Y.Qi, "Selection of the Optimal Number of Topics for LDA Topic Model-Taking Patent Policy Analysis as an Example," Entropy (Basel). 3;23(10):1301. 2021, doi: 10.3390/e23101301.

[27] A.Elsaadawy, M.Torki and N.Makky, "A Text Classifier Using Weighted Average Word Embedding," 2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC), Alexandria, Egypt, pp. 151-154, 2018. doi: 10.1109/JEC-ECC.2018.8679539.

[28] J.Brownlee, A Gentle Introduction to k-fold Cross-Validation, https://machinelearningmastery.com/k-fold-cross-validation/ (Last access: 20/9/2023).